In the Claims:

Please cancel Claims 1-11; and add new Claims 12-31, all as shown below. Applicant respectfully reserves the right to prosecute any originally presented or canceled claims in a continuing or future application.

1-11. (Canceled).

12. (New) A system for interleaving resource enlistment synchronization, comprising:

an application server with a plurality of threads, running on one or more processors;

a transaction manager, wherein the transaction manager maintains an enlistment data structure to manage resource object enlistment in one or more transactions;

one or more resource objects, wherein each resource object is wrapped with a wrapper object, wherein the transaction manager uses the wrapper object to synchronize concurrent enlistment requests;

wherein, after receives a request from one of the plurality of threads to enlist one of the one or more resource objects, the transaction manager

first checks to see if there is a lock being held on the resource object by another thread;

if not, grants a lock to an accessor associated with the thread and holds the lock until an owner of the thread delists the resource object.

13. (New) The system of Claim 12, wherein:

the transaction manager maintains a collection of wrapper objects that wraps the one or more resource objects.

14. (New) The system of Claim 13, wherein:

the collection of wrapper objects is periodically processed to remove objects that are unused or no longer active.

15. (New) The system of Claim 12, wherein:

each of the one or more resource objects resides in a server node.

16. (New) The system of Claim 12, wherein:

the transaction manager signals to one or more waiting threads once a lock is free.

17. (New) The system of Claim 12, wherein:

the transaction manager use a priority method to determine which thread will be granted a lock.

18. (New) The system of Claim 12, wherein:

after obtains a lock, the thread uses the wrapper object to initiate work on the resource object.

19. (New) The system of Claim 12, wherein:

the wrapper object receives a delist call from the transaction manager and send an end call to the resource object to end work performed by the resource object associated with the thread and release the lock on the resource object.

20. (New) The system of Claim 12, wherein:

once the transaction manager enlists the resource object and obtains a lock to the resource object, any attempted enlist from a second thread is blocked.

21. (New) The system of Claim 12, wherein:

the enlistment data structure maintains a mapping of resources and transaction identification information in use.

22. (New) A method for interleaving resource enlistment synchronization, comprising:

providing an application server with a plurality of threads, running on one or more processors;

maintaining an enlistment data structure, at a transaction manager, to manage resource object enlistment in one or more transactions;

wrapping each of one or more resource objects with a wrapper object, wherein the transaction manager uses the wrapper object to synchronize concurrent enlistment requests;

receiving a request from one of the plurality of threads to enlist one of the one or more resource objects at the transaction manager;

first checking to see if there is a lock being held on the resource object by another thread;

if not, granting a lock to an accessor associated with the thread and holding the lock until an owner of the thread delist the resource object.

23. (New) The method of Claim 22, further comprising:

maintaining a collection of wrapper objects that wraps the one or more resource objects.

24. (New) The method of Claim 23, wherein:

the collection of wrapper objects is periodically processed to remove objects that are unused or no longer active.

25. (New) The method of Claim 22, further comprising:

signaling to one or more waiting threads once a lock is free.

26. (New) The method of Claim 22, further comprising:

using a priority method to determine which thread will be granted a lock.

27. (New) The method of Claim 22, wherein:

after obtains a lock, the thread uses the wrapper object to initiate work on the resource object.

28. (New) The method of Claim 22, wherein:

the wrapper object receives a delist call from the transaction manager and send an end call to the resource object to end work performed by the resource object associated with the thread and release the lock on the resource object.

29. (New) The method of Claim 22, wherein:

once the transaction manager enlists the resource object and obtains a lock to the resource object, any attempted enlist from a second thread is blocked.

30. (New) The system of Claim 12, wherein:

the enlistment data structure maintains a mapping of resources and transaction identification information in use.

31. (New) A computer-readable storage medium, storing instructions for interleaving resource enlistment synchronization, the instructions comprising the steps of:

providing an application server with a plurality of threads;

maintaining an enlistment data structure, at a transaction manager, to manage resource object enlistment in one or more transactions;

wrapping each of one or more resource objects with a wrapper object, wherein the transaction manager uses the wrapper object to synchronize concurrent enlistment requests;

receiving a request from one of the plurality of threads to enlist one of the one or more resource objects at the transaction manager;

first checking to see if there is a lock being held on the resource object by another thread;

if not, granting a lock to an accessor associated with the thread and holding the lock until an owner of the thread delist the resource object.